

Carnegie Mellon University
Software Engineering Institute

Report of the Reuse and Product Lines Working Group of WISR8

Paul Clements

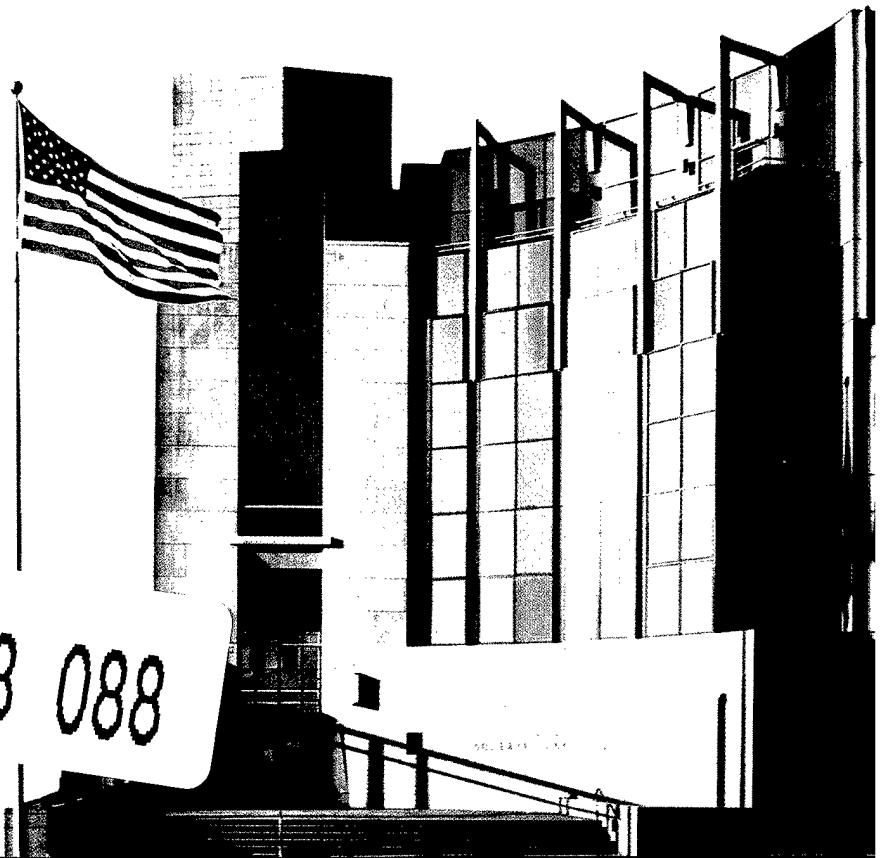
August 1997

SR

SPECIAL REPORT
CMU/SEI-97-SR-010

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

19970828 088



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of, "Don't ask, don't tell, don't pursue," excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Special Report
CMU/SEI-97-SR-010

August 1997

Report of the
Reuse and Product Lines
Working Group of WISR8



Paul Clements

Product Line Systems

Unlimited distribution subject to the copyright.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

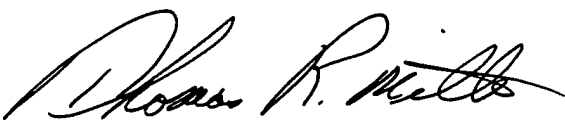
DTIC QUALITY INSPECTED 4

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright © 1997 by Carnegie Mellon University.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

Requests for permission to reproduce this document or to prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through SAIC/ASSET: 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 / FAX: (304) 284-9001 / World Wide Web: <http://www.saic.com/-contact.html> / e-mail: webmaster@cpqm.saic.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218. Phone: (703) 767-8274 or toll-free in the U.S. — 1-800 225-3842).

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

1	Introduction	1
2	Working Group Charter	1
	2.1 Scope	1
	2.2 Goals	1
	2.3 Process	2
3	Working Group Members	2
4	Issues Addressed	2
5	Terminology	3
6	Experience Reports	4
7	Discriminating Factors	5
	7.1 Organizational Factors	6
	7.2 Environmental Factors	6
8	Risks	7
9	The Role of Architecture and Architects	7
10	Managing Change and Variation	9
11	Miscellaneous	10
12	Summary	10
	References	13

List of Figures

6-1. An Organization for Product Lines

5

Report of the Reuse and Product Lines Working Group of WISR8

Abstract: This report summarizes the discussions held by the Reuse and Product Lines working group at the Eighth Workshop on Software Reuse (WISR8). The working group was chartered to explore the range of issues and practices necessary for the successful fielding of a software product line, which is a collection of systems that are built from a common set of core assets. Issues addressed include the relation between a product line and a domain, benefits of the product line approach to software development, organizational factors, effects of the products' domain and context, risks, the role of architecture and architects, and managing change and variation. Maintaining intellectual control was a theme that arose repeatedly during the discussion.

1 Introduction

This report synthesizes the discussions held at the Eighth Workshop on Software Reuse (WISR8) by the Reuse and Product Lines working group.

2 Working Group Charter

This sections contains the charter for the Reuse and Product Lines working group.

2.1 Scope

A product line is a group of related systems that, taken together, address a specific market segment. Product lines are most efficiently built from a common set of reusable assets, such as domain models, requirements, architectures, software components, test plans, project schedules, tools, and environments. A product line built in this way epitomizes reuse in a planned and systematic way.

This working group will explore the issues associated with successful fielding of a product line from a common asset base. We will try to understand the differences between planned, systematic reuse of an asset base over which the developing organization has broad control, and reuse that occurs serendipitously or opportunistically or is based upon assets built by others.

2.2 Goals

The proposed working group will address practical issues of systematic reuse in a product line context. Actual experience, where available, will take precedence over theoretical supposition. Topics may include (but are not limited to) the following:

- organizational issues: how best to organize to field a product line
- configuration control issues and associated tool support

- techniques (such as parameterization) and issues (such as testing of the multitude of resulting configurations) associated with making software components tailorable
- cost models for product lines
- leveraging test plans and test cases across all members of a product line
- product lines and application generators
- leveraging design and coding reviews across all members of a product line

2.3 Process

Experience reports will be sought, and participants will be asked to choose a small list of topics, which will then be examined to the fullest extent possible under the time constraints.

As a stimulus for discussion, the leader will present a case study in successful product line development (performed by a Swedish defense contractor called CelsiusTech Systems AB). CelsiusTech routinely turns out million-line-plus, hard real-time Ada systems for shipboard command and control for a wide variety of different ships around the world using the product line approach.

3 Working Group Members

Members of the working group included

- Jeff Poulin (Lockheed Martin Federal Systems)
- Cuauhtemoc Lemus (University of Houston)
- Greg Butler (Concordia University)
- David Hale (University of Alabama)
- Fatima Mili (Oakland University)
- Cornelia Boldyreff (University of Durham)
- Oh-Cheon Kwon (University of Durham)
- Ernesto Guerrieri (Buzzeo, Inc.)

The working group leader was Paul Clements of the Software Engineering Institute.

4 Issues Addressed

The working group addressed the following issues, each of which will be treated in detail in subsequent sections:

- **Terminology:** What is the meaning of product line, product family, and domain? How do these terms and concepts relate to each other?
- **Experience reports:** Four members recounted their experience with (aspects of) product line development to provide concrete examples of the concepts and issues in actual practice.

- **Key organizational factors:** What are the factors that affect an organization's transition to a product line approach? What factors are relevant with respect to the organization itself and the products that it builds?
- **Risks:** Product lines are not without their downside, and we discussed some of the major drawbacks with the approach.
- **Architecture:** What is the role of architecture in a product line? What responsibilities are borne by the architect(s)?
- **Managing change and variation:** What happens when a variant is required that is outside the scope of the original product line?

5 Terminology

The group first approached the issue of terminology and product lines. We agreed to distinguish between a *product line*, which is a set of related systems that together address a market segment, and a *product family*, which is a set of related systems that are built from a common set of core assets. Thus, a product line is a market- or customer-driven concept, whereas a product family is a technology- or implementation-dependent concept.

A product line need not be built as a product family (i.e., from a common set of core assets), although in the context of a reuse conference we agreed to consider this type of product line. Conversely, a product family need not be a product line (i.e., addressing a particular market niche).

Questions about the definition of *domain* soon arose. What is the difference between a product line and a domain? One participant maintained that there was no difference, except that domain was a term familiar to developers whereas product line resonated with management, and so the choice of term was audience-dependent. However, borrowing a definition from the domain analysis working group, we agreed that a *domain* was a specialized body of knowledge—an area of expertise. Thus, it is an intangible thing, unlike a product line or product family, which are specific sets of actual (or envisioned) products.

Lingering confusion was cleared when we compared product lines and domains. There are

- product lines that encompass more than one domain. An automobile product line requires expertise in engines, suspension, interior design, and the like.
- product lines that are synonymous with domains. "Relational databases" describes both a body of knowledge (domain) and a set of products (product line).
- product lines that form a subset of a domain. Cellular phones form a product line wholly contained (in addition to other product lines such as pagers) within the domain of wireless communication.

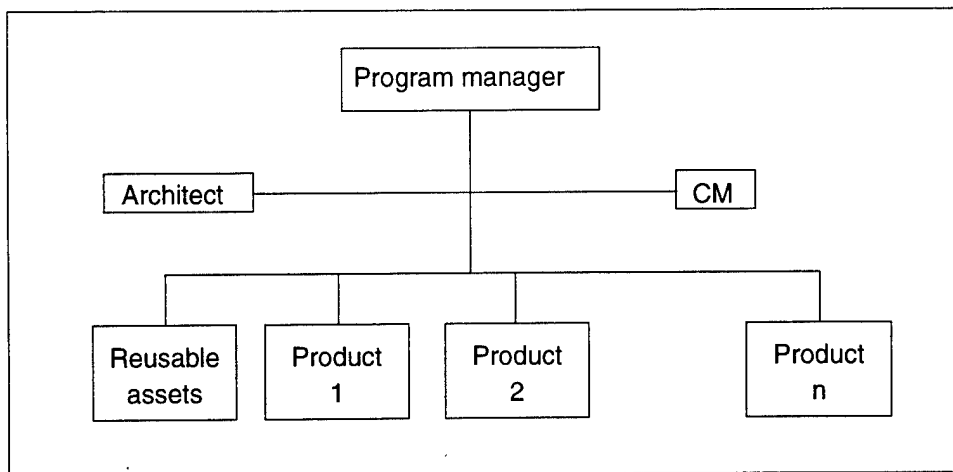
Further, the meanings of these terms are depend upon the context in which they are used. One organization's product (e.g., a GUI widget) may be but a single component in the product of an organization building large systems. Either may be built in a product line. Similarly, a product that grew out of a carefully planned product line development (which we referred to

variously as a “top-down,” “a priori,” or “planned” product line) could serve as a component in a product line where the components are not designed from scratch but rather found opportunistically (“bottom-up,” “a posteriori,” or “unplanned”).

6 Experience Reports

Paul Clements presented the CelsiusTech case study [Brownsword 96]. CelsiusTech is a Swedish company that builds shipboard command and control systems for many navies around the world. In December of 1985, they landed two major contracts simultaneously. Realizing that they could not possibly complete those contracts one at a time, they adopted a product line strategy as the only way to fulfill their company’s obligations. Using their extensive domain knowledge from many years and many products, they produced an architecture that would serve these two systems as well as future systems that were anticipated. Today, CelsiusTech has fielded over 55 variations of their basic system on several different classes of ships, including submarines. A typical system has 15-30 nodes on a local area network, 30-70 CPUs, and 100-300 separate Ada programs of 1-1.5 million lines of Ada code. Their architecture is a carefully layered one, with application-dependent, machine-independent software at the top and application-independent, machine-dependent software at the bottom. Strict and careful attention was paid to design principles of abstraction, encapsulation, and information-hiding. The components that are reused from product to product are large-scale pre-integrated chunks called system functions, which are user-visible features of the systems; this large-grained reuse was a key to their success. CelsiusTech reports around 80% verbatim reuse from system to system, delivery times an order of magnitude less than before, and whole systems integrated by one or two people. Today they field systems in their product line using a work force about a third the size of a few years ago, and report greater reliability, predictability, and customer satisfaction. One of the effects of product line development for CelsiusTech has been a different customer interaction model. Customer requirements are now negotiated with the core asset group (to make sure the proposed system does not radically depart from the product line); customers have formed a user group to drive their requirements evolution jointly.

Jeff Poulin presented an organization for a management information system product line at Lockheed Martin. It resembled the CelsiusTech organization in that there was a group responsible for maintaining the reusable assets, and this group was separate from the groups responsible for fielding individual products. An architect oversees both the reusable asset group and the product groups to assure that products conform to the architecture and to evolve the architecture to respond to change. A separate group is responsible for configuration management, maintaining a small number of versions of each of the assets, as illustrated by Figure 6-1. Jeff reported around 20% reuse, which is quite high for MIS systems.



6-1. An Organization for Product Lines

Ernesto Guerrieri explained his company's approach to product lines is to deliver product "kits" to customers. The domain is software for institutions of higher education, e.g., student information systems, alumni development, financial management, and human resources. A kit includes all of the documentation and configurable software with which a customer can produce a product to satisfy his or her particular requirements. Domain analysis forms an explicit front end to the process of producing the kits (which is equivalent to defining the product lines). The early or "premier" customers got to participate in the domain analysis effort, thus assuring applicability to their particular needs, in return for being the company's "early adopters." Using this product line approach, the company can deliver systems to customers within about a month.

Cornelia Boldyreff and Oh-Cheon Kwon presented a process model for configuration management in a product line environment.

From these four presentations, some important themes emerged:

- Deep knowledge of the application domain is critical.
- It is important that an organizational entity is assigned responsibility for maintaining the reusable assets, and that this entity is separate from the entities charged with producing a specific product in the product line.

Other themes also emerged, such as the key role of architecture, which will be discussed in the following sections.

7 Discriminating Factors

Different organizations will take different paths towards successful development and deployment (marketing) of a product line. The group spent time trying to enumerate what the discriminating factors were that would affect an organization's approach or chances of success.

These factors are related to the organization itself, and the context in which the organization operates (such as the nature of its products and its market).

7.1 Organizational Factors

Organizational factors include the following:

- Whether or not the organization has sufficiently deep domain knowledge.
- Whether or not an empowered champion exists for the transition to the product line approach.
- Whether or not the organization possesses people with strong architectural skills with the ability to grasp abstractions at the component and the product level.
- What legacy assets are available, e.g., software components, documentation, test plans and cases, tools, development environments. (Legacy assets could help or hinder, depending on how applicable they are; they might speed the process, or they might inhibit new development necessary to accommodate the variation that the product line needs to support. They might also represent old, established systems that must continue to be supported.)
- The source of the funding to pay for the product line development: whether previous projects have provided a windfall, whether R&D money is available, or whether the new (product line) contracts are fixed-price or cost-plus.
- Whether or not there is a particular group, apart from groups responsible for individual products, that maintains the reusable assets for the product line.

7.2 Environmental Factors

Environmental factors describe the product, what it does, and how it is built:

- Whether the reuse from product to product is large-grained or small-grained. With small-grained reuse typically comes the need to search a library, evaluate potential choices, and choose from a long list of candidate components. With large-grained reuse, systems are assembled from a small number of components. Adaptation and composition are the only steps.
- How challenging is it to build each individual product? For instance, are the products hard real-time or highly distributed? The harder it is to build a product, the more advantageous a product line may be—but the risk is higher.

Environmental factors also describe the product's domain:

- Whether the domain is stable or not, meaning how often variations are expected and how wide the variation across the products is. Is 80% reuse expected, or 20%? Will new products expand the product line (i.e., vary in new ways) or be contained within it (i.e., vary in ways accounted for by the components' variability capabilities)? Will newer products use the same or different technology?
- What is the life cycle of the products? Will older versions have to be maintained and supported for 20 years, or can they be replaced by current versions?
- Are there stable, well-defined architectures for the domain?

- How many products are likely to populate the product line? Will a few major versions be maintained, or many?

Finally, environmental factors describe the marketplace:

- What are the products' requirements for time to deployment? Government contracts often set relatively far, and fixed, delivery dates, whereas in the open market there is a premium on getting the product on shelves as soon as possible.
- How big is the customer base? Will the product line consist of large, expensive systems purchased by a few customers with whom close relationships can be formed, or is the product line for the mass market with thousands of anonymous consumers?

8 Risks

Risks associated with product line development identified by the group include the following:

- An organization can come to rely on its product line infrastructure to accommodate new requirements. But product lines are only flexible to a point. Organizations must guard against complacency in case a competing organization enters the market with an innovative technology outside the scope of the product line. It is important in any organization to keep looking to the horizon, but especially so in a product line organization. The product line becomes its paradigm, and it becomes vulnerable to a paradigm shift that may catch it by surprise.
- Similarly, a product line supports variability, but only within certain dimensions. Competitors may offer different systems in the same domain that offer more variability, the mere novelty of which may lure customers away. General Motors lost market share in the 1980s because its products were too similar to each other—their different automobiles were perceived as the same cars with different nameplates.
- Experience has shown that cost and time to market of early products in a product line may be higher than normal.
- The first product, aiming to be general, may not meet the first customer's requirements well. Or, if the organization fixates on the first system in an attempt prove the concept by making the first system work at any price, that first system may lead to an approach that is not general enough.
- There are additional costs of training and infrastructure setup. Bringing a new person aboard may be more complicated and require more methodological training.

9 The Role of Architecture and Architects

A product line cannot exist without an architecture. Usually the architecture is explicitly designed for the product line. It is possible that the architecture for a single product will be general enough to support all future (less general) versions of the system, but this is highly unlikely.

The architecture forms the basis for product line evolution for growing (as opposed to building) software [Brooks 87]. It is an expression of the commonality across products assumed by the product line designer. It is also at the very least a de facto expression of the variability that the product line encompasses. Every architecture divides all changes into three classes: (a)

changes that are confined to a single component; (b) changes that affect several components; and (c) changes that affect the architectural underpinnings. Hence, an architecture that is successful in the context of a product line is one in which the variation across the products falls into the first or second class. Further, an architecture should be built so that the cost of a change (producing a variant) is proportional to the value of making that change and the likelihood that the change (variant) will be required.

In addition to the standard notion of an architect responsible for the overall structuring of the system, its parts, and their interaction and coordination, there are also domain architects. These are experts in particular domains, such as security or networking, who may be called in to provide solutions to particular problems. Hence, the overall architecture should accommodate the work of these experts by providing them with well-encapsulated sections of the system in which to apply their solutions.

Architectures are often represented by system views, reflecting the different structures that are present in every system. These different structures and the views that show them provide opportunities for managing different aspects of a system's quality attributes such as performance or modifiability. Views most often used include the

- modular or logical view. This structure decomposes the system into work assignments assailable by different teams.
- process view. This structure decomposes the system into processes or tasks that execute concurrently and synchronize with each other.
- physical view. This structure allocates software to processors.
- functional view. This structure decomposes the system into (possibly overlapping) chunks of software, where each implements a particular user-visible function of the system.

Performance is most affected by the process and physical views; modifiability is usually the purview of the modular view; correctness often resides in the functional and process views.

How might each view change in a product line context, as opposed to a single-system development? If some products were proper subsets of others, that subsetting might be accomplished by eliminating processes and/or modules and/or functions, thus influencing all affected views. Some versions might run on different hardware configurations, thus affecting the physical view.

Ownership of the architecture is a key issue. Ownership means controlling how it evolves over time, as well as making sure that "architectural drift" does not occur—this is when individual projects make unauthorized changes to the reusable components or how they interact with each other.

This implies the importance of having a central architectural authority to hold tightly to the reins of the product line. But in very large systems, it is unlikely that a single person can perform all of the duties, attend all of the meetings, keep abreast of all anticipated changes, perform all of the enforcement, and maintain all of the designs that are necessary to sustain a product line throughout its life. Hence, a hierarchical arrangement is the norm, where there is a chief archi-

tect who retains authority, but who has a staff that helps promulgate policy and brings important matters to his or her attention. Here, the architect can help his or her own plight by making sure that the architecture is cleanly divided into well-defined components with well-specified interfaces. Issues brought to the architect's attention will often be couched in terms of those interfaces; hence, the software-to-software interfaces are also the backbone of the team-to-team or architect-to-component interfaces as well.

10 Managing Change and Variation

Configuration management and version control are key issues and must be addressed carefully and rigorously in order for the product line to succeed. A corollary issue is that it is important to decide how many versions of reusable components are going to be maintained and supported. CM is more complicated in a product line context for two reasons: (1) a change must be considered not from the point of view of a single product, but in terms of keeping the changed component usable by all of the products that currently employ it; and (2) it is more likely to be necessary to maintain separate versions of reusable components, as opposed to simply supplying the most recent one, as may suffice in one-at-a-time development.

Strong, centralized architectural control is key to product line development, but so is management of change and evolution. In conventional development, the architect answers to a single customer or set of customers, and their needs are often paramount. But in a product line, the architect answers to users of all versions of the system, and keeping the product line intact is more important than making changes to accommodate a single user's needs.

Variation in the reusable assets is often handled by parameterization. However, this approach carries its own problems. New combinations of parameters effectively produce new versions of the code that must be tested. However, regression testing tools do not know the difference between a parameter whose new value makes a difference and one that does not, and testing is sometimes overdone. Further, some combinations of parameters may be illegal, and these must be identified and ruled out. This is analogous to the feature interaction problem in telecommunications.

What happens when a request for a variant is outside the boundaries of the product line? Rejection or renegotiation of the request is certainly a possibility, but often this is not the desirable response. Impact analysis can be done to gauge the extent (and cost) of the change. As noted in the previous section, the product line architecture will determine how hard it will be to make the required modification(s). A business analysis can be done to determine the value of the new variant, perhaps in the context of anticipated market shifts where the variant might be of value beyond the specific customer who requested it. Then, as before, the cost and value can be compared. Thus, "outside the product line" is a phrase with no operational meaning. Ease of change is a spectrum, not a boolean. The required change(s) will be quite easy (e.g., via parameterization that is already in place), somewhat hard (e.g., a component will have to be replaced), or very hard (e.g., the entire architecture must change).

When a new product is requested, who is responsible for providing it? An individual product manager may well be the one who gets the request from his current customer. It may be to his advantage to let the reuse group handle the request because it will cost him less; however, it may take longer that way, thus alienating his customer. The answer to this question may well depend on two things: (a) What part of the organization fields the new request? Does a product manager, or is there a product-line-level clearinghouse for such requests? This is a customer management issue. (b) How large-grained is the reuse in the product line? Products with 10% reuse may be more likely to handle changes themselves, as opposed to projects with 80% reuse consisting mainly of a few very large reusable chunks.

We conjectured that the ability to accommodate variation was the primary difference between pure-hardware product lines and software product lines. Software is more variable, and its variations can typically be bound much later (even at runtime) than with hardware.

11 Miscellaneous

We did not thoroughly address every relevant topic. This section records those issues that we touched on only briefly. They included

- the necessity of having a strong, well-defined process to control the production of individual systems (products) using the reusable assets, as well as to control the evolution of those assets.
- the question of how to price an individual product in a product line, and how to amortize the high up-front investment typically incurred in product line development.
- the observation that organizations change in the face of a real, perceived, or anticipated crisis. How does the product line champion, who already perceives the crisis that is causing him to advocate change, get his company to see the crisis? How does he crystallize his vision? Risk management is one approach that many organizations are adopting as a way to spot crises in the making, and this may be a vehicle for bringing the situation to management's attention.
- noting that customer management is important, both in terms of keeping new requirements in check (in line with the variations supported by the product line), as well as working to keep their expectations in line, and making them understand that the easy variation may only come once the product line is mature and not when the early products are being produced.

12 Summary

Intellectual control seems to be a key to successfully developing a product line. An organization must achieve intellectual control over the domain(s) involved, as well as over its goals, plans, assets, and capabilities. Product lines strongly resemble conventional system building in many ways, especially when building a system with changeability in mind, and when building a system using reusable assets. However, there are important differences having to do with process, organizational structure, customer interfaces, and technical issues. An organization

must be aware of these key issues if it hopes to successfully develop a product line and enjoy its considerable benefits.

References

- [Brooks 87] Brooks, F. P., Jr. "No Silver Bullet: Essence and Accidents of Software Engineering." *Computer* 20, 4 (April 1987): 10-19.
- [Brownsword 96] Brownsword, Lisa & Clements, Paul. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, 1996.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE August 1997	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Report of the Reuse and Product Lines Working Group of WISR8	5. FUNDING NUMBERS C — F19628-95-C-0003	
6. AUTHOR(S) Paul Clements		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-97-SR-010	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/AXS 5 Egin Street Hanscom AFB, MA 01731-2116	10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES		
12.a DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS	12.b DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) This report summarizes the discussions held by the Reuse and Product Lines working group at the Eighth Workshop on Software Reuse (WISR8). The working group was chartered to explore the range of issues and practices necessary for the successful fielding of a software product line, which is a collection of systems that are built from a common set of core assets. Issues addressed include the relation between a product line and a domain, benefits of the product line approach to software development, organizational factors, effects of the products' domain and context, risks, the role of architecture and architects, and managing change and variation. Maintaining intellectual control was a theme that arose repeatedly during the discussion.		
14. SUBJECT TERMS domain, product families, software architecture, software product lines, strategic reuse		15. NUMBER OF PAGES 14
		16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED
20. LIMITATION OF ABSTRACT UL		